CSC230: Extra Credit
Name: Anibal Ruiz, N00866307
Prof: Steve Ochani.

CONCLUSIONS:

Output:

Merge 20000 Average Time: 3.933333333333333
Quick 20000 Average Time: 1.6666666666666667
Selection 20000 Average Time: 105.06666666666666
Bubble 20000 Average Time: 554.8
****
Merge 20000 Worst Time: 11.0
Quick 20000 Worst Time: 5.0
Bubble 20000 Worst Time: 577.0
Selection 20000 Worst Time: 151.0
****
Merge 20000 Big O: 72649.38498075992
Quick 20000 Big O: 171452.54855459338
Selection 20000 Big O: 3807106.598984772
Bubble 20000 Big O: 720980.5335255949
****
Merge 40000 Average Time: 8.062222222222223
Quick 40000 Average Time: 3.1111111111111116
Selection 40000 Average Time: 407.6044444444444
Bubble 40000 Average Time: 2202.1200000000003
****
Merge 40000 Worst Time: 11.0
Quick 40000 Worst Time: 5.0
Bubble 40000 Worst Time: 2186.0
Selection 40000 Worst Time: 413.0
****
Merge 40000 Big O: 75848.62812345372
Quick 40000 Big O: 196556.3020227786
Selection 40000 Big O: 3925374.273533164
Bubble 40000 Big O: 726572.575518137
****
Merge 60000 Average Time: 8.80414814814815

Quick 60000 Average Time: 4.740740740740741
Selection 60000 Average Time: 922.9736296296297
Bubble 60000 Average Time: 5039.674666666667
****

Merge 60000 Worst Time: 11.0
Quick 60000 Worst Time: 5.0
Bubble 60000 Worst Time: 4950.0
Selection 60000 Worst Time: 918.0
****

Merge 60000 Big O: 108171.79320370188
Quick 60000 Big O: 200888.54145342484
Selection 60000 Big O: 3900436.463655636
Bubble 60000 Big O: 714331.8245939685
****

Merge 80000 Average Time: 11.786943209876544
Quick 80000 Average Time: 6.649382716049383
Selection 80000 Average Time: 1723.7315753086418
Bubble 80000 Average Time: 9171.37831111111
****

Merge 80000 Worst Time: 14.0
Quick 80000 Worst Time: 7.0
Bubble 80000 Worst Time: 9452.0
Selection 80000 Worst Time: 2041.0
****

Merge 80000 Big O: 110547.49031726299
Quick 80000 Big O: 195960.594540439
Selection 80000 Big O: 3712875.0738664465
Bubble 80000 Big O: 697823.1387801777
****

Merge 160000 Average Time: 23.652462880658437
Quick 160000 Average Time: 13.776625514403293
Selection 160000 Average Time: 6542.848771687243
Bubble 160000 Average Time: 35437.358554074075
****

Merge 160000 Worst Time: 27.0
Quick 160000 Worst Time: 15.0
Bubble 160000 Worst Time: 35104.0
Selection 160000 Worst Time: 6852.0
****

Merge 160000 Big O: 116944.86086646852
Quick 160000 Big O: 200777.32227213823
Selection 160000 Big O: 3912668.761469536
Bubble 160000 Big O: 722401.4724725267
****

Merge 320000 Average Time: 51.043497525377234
Quick 320000 Average Time: 29.11844170096022
Selection 320000 Average Time: 26210.523251445815
Bubble 320000 Average Time: 142041.35723693826
****

Merge 320000 Worst Time: 59.0
Quick 320000 Worst Time: 29.0
Bubble 320000 Worst Time: 140832.0
Selection 320000 Worst Time: 26575.0
****

Merge 320000 Big O: 114648.64762737621
Quick 320000 Big O: 200974.62706127038
Selection 320000 Big O: 3906827.765994769
Bubble 320000 Big O: 720916.7948824033
****

Merge 640000 Average Time: 129.93623316835848
Quick 640000 Average Time: 61.14122944673068
Selection 640000 Average Time: 104356.7015500964
Bubble 640000 Average Time: 566257.0238157959
****

Merge 640000 Worst Time: 325.0
Quick 640000 Worst Time: 68.0
Bubble 640000 Worst Time: 563564.0
Selection 640000 Worst Time: 108176.0
****

Merge 640000 Big O: 95001.4912847084
Quick 640000 Big O: 201895.4482043329
Selection 640000 Big O: 3924999.4865291105
Bubble 640000 Big O: 723346.4359344413
****

MATHEMATICAL RESULTS:

Avg:
For each sort algorithm, you have to run 15 times for each size (20000, 40000…) To get the average, you have to add each time, (stop - start) in millis, the algorithm takes to sort the numbers from 1 to the size of array. That avg have to be divided by 15, the number of repetitions for each algorithm.

Worst Time:
For each time any algorithm sorts the numbers, you compare a variable worst for each sorting to see if it is less than the current time , if it is less then changed that value to the worst variable.

Big O:
For Bubble and Selection: O(n^2), n is the size of the array or the amount of numbers to be sorted to the second.
For merge and quick: 0(n log n), n is the size of the array or the amount of numbers to be sorted.
Each answer has to be divided by the avg time.

The results for big O for bubble and selection sorts must be the worst because they are inefficient. 0(n^2)

Stack overflow: Because the pattern repetition of code caused by the recursion, the stack used to store memory overflows.
SOmetimes you have to inspect your recursion line by line and understand why the recursion never ends. So the best way to avoid stack overflow is not to use recursion at all. To fix it, it's better to do iteration code.

Heap out of memory: Java is allowed to use a limited amount of memory. One of the memory regions is the heap memory. Based on a research i did in the GeeksofrGeeks and Plumbr pages,, this error will occur when the application attempts to add more data into heap space area, but there is not enough room for it. What is the solution? If the amount of heap you have in your JMV is not enough, then you can allocate more heap. If we wish to solves this problem without changing the heap space, we have to figure out wich part of our code is being responsible for the allocation of most memory. Like the stack overflow, recursion takes a lot repeated memory, so stop coding recursive would prevent us to blow up the memory spaces.